Working with the Natural Web Interface

This section covers the following topics:

- Setting up your Environment
- Building Subprograms in Natural

Setting up your Environment

Prerequisites on the Web Environment Side

The following software must be installed:

On the web client:	Browser software, such as Netscape Navigator or Microsoft Internet Explorer.
On the web server:	HTTP server software, such as Netscape Fast Track Server, Microsoft Internet Information Server or Apache Server.

Middleware Prerequisites

Different prerequisites must be met if communication is to be used by RPC or DCOM:

RPC	The broker of the Software AG product EntireX must be installed (for installation information, see EntireX documentation).
DCOM	On non-Windows platforms, DCOM is not a part of the operating system. The Software AG product EntireX contains a DCOM implementation for several platforms and must be installed (for installation information, see EntireX documentation).

The Natural Web Server Extensions part is needed for communication between a web browser and a Natural DCOM/RPC server.

Prerequisites on Natural Server Side

The following prerequisites must be met:

- Natural Version 2.3.1/3.1.1 (Mainframe), 4.1.1 (UNIX / NT) or above must be installed.
- The library SYSWEB (the Natural Web Interface’s server programs and server-program online documentation). Either Natural steplibs must be available or the contents of the library SYSWEB must be copied to the library SYSTEM or to the user library that will be called by the RPC.
- The global data area W3CONTEX.
- The parameter data Area W3PARM.
- The Natural RPC stub or NaturalX.

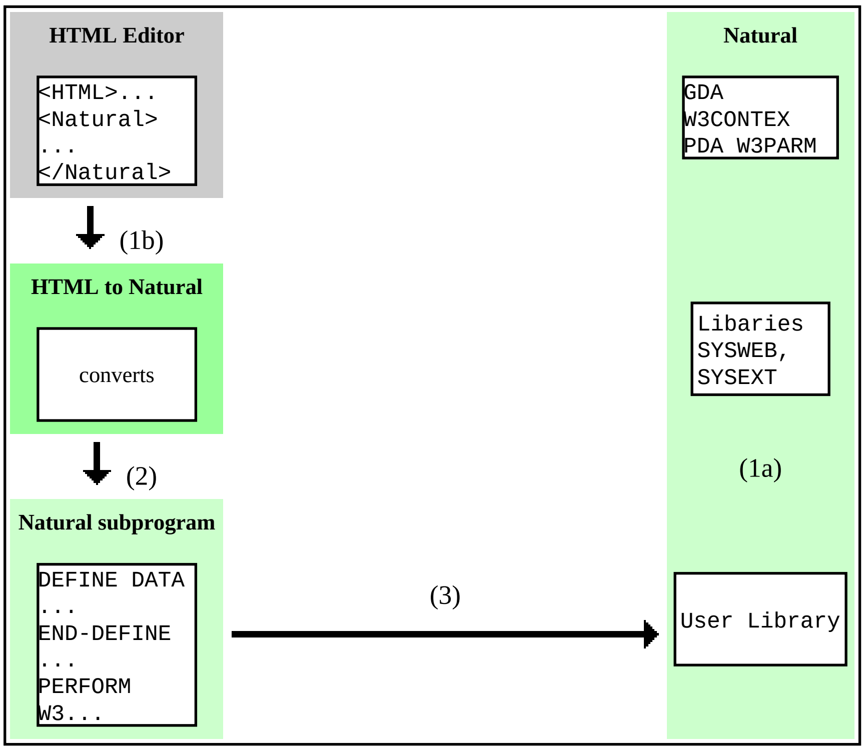
Building Subprograms in Natural

The following diagram illustrates how you can build a subprogram:

1. Using a HTML editor
2. You use a HTML editor to enter HTML and Natural code
3. Then convert it to Natural source.

4. Finally move the generated program to Natural. (You code directly in Natural.)

Each stage of the process is identified by a number; what happens at these stages is explained below.



1. 1a. Natural Code is written and stored in User Library.
   You write Natural code on the server side either by including HTML tags in the code or by calling pre-fabricated subprograms that generate HTML tags. Then you store it as a server program or use the subprogram WEB-WIZ to generate a default program.

   1b. Natural Code is entered as HTML. Continue with 2.
   You use an HTML editor to create HTML pages.

2. Program HTML2NAT generates Natural Sources out of HTML.
   You start the program HTML2NAT out of the library SYSWEB and let it convert your HTML pages created in step 1b.
3. Generated Natural Source is moved to the User Library.
   You move the generated Natural sources to your server environment by using the Natural SYSUNLD function.

Before You Write Your Subprograms

Keep the following things in mind:

- The returning HTML page is limited to the maximum data that can be transmitted. This maximum is determined by the return page variable.
- You must initialize and end the access to the Natural server subroutines by calling the subroutines W3INIT and W3END in the library SYSWEB.
- Always use the parameter data areas W3PARM and W3CONST.
- Use the subprogram WEB-WIZ to generate a frame (default program) for your own program.

Ways to Create Your Subprograms

There are two basic alternatives:

1. Coding in Natural direct
2. Using an HTML editor.

1. Coding In Natural Direct

There are two alternatives:

- Entering calls to SYSWEB subroutines (such as W3HTML or W3TEXT) for your return page in the program editor (see the programs in the library SYSWEB, which help you perform only basic system functions; this approach requires a good knowledge of the data type you are creating, for example HTML or XML); or
- calling subprograms that generate HTML tags (see the library SYSWEB; the programs in the library SYSWEB enable you to perform basic system functions and in addition, the programs in the library SYSWEB generate HTML tags; this approach requires less explicit HTML knowledge and you can still modify the programs you are calling).

Example: Entering Calls to SYSWEB Subroutines in the Program Editor

```
*
* Example E3END
*
DEFINE DATA
PARAMETER USING W3PARM
LOCAL USING W3CONST
LOCAL
1 W3VALUE        (A250)
END-DEFINE
* --- ERROR HANDLING ---
ON ERROR
  PERFORM W3ERROR ##W3ERROR
  PERFORM W3END ##RPC
  ESCAPE ROUTINE
END-ERROR
*
* --- INITIALIZE W3 PROCESSING ---
PERFORM W3INIT ##RPC
*
* --- SET TYPE OF RETURN-PAGE ---
PERFORM W3CONTENT-TYPE 'text/html'
* --- WRITE THE DOCUMENT ---
PERFORM W3TEXT '<HTML><BODY><H2>Initialize</H2>'
*
* --- END THE HTML PAGE ---
COMPRESS '<HR>generated:' *DATE *TIME ##HTTP_NEWLINE
         '</BODY></HTML>' ##HTTP_END INTO W3VALUE
PERFORM W3TEXT W3VALUE
*
* --- END W3 PROCESSING ---
PERFORM W3END ##RPC
*
END
```

Example: Calling Subprograms that Generate HTML Tags

```
*
* Example E3IMAGE
*
DEFINE DATA
PARAMETER USING W3PARM
LOCAL USING W3CONST
LOCAL
1 H3VALUE            (A250)
1 H3VALUE-MAX        (I004)
1 H3URL              (A250)
*
1 II                 (I001)
1 GIF                (A064)
END-DEFINE
* --- ERROR HANDLING ---
ON ERROR
  PERFORM W3ERROR ##W3ERROR
  PERFORM W3END ##RPC
  ESCAPE ROUTINE
END-ERROR
*
* --- INITIALIZE W3 PROCESSING ---
PERFORM W3INIT ##RPC
*
```

```
* --- Pathname of picture ---
PERFORM W3READ-ENVIRONMENT "PICTURES" ' ' H3VALUE H3VALUE-MAX
IF H3VALUE-MAX EQ 0 THEN
  GIF  := "/pictures"
ELSE
  GIF  := H3VALUE
END-IF
*
* --- START HTML API ---
PERFORM H3-OPEN-HTML 'HTML Api -Image' " " " "
* --- THE LEVEL 2 HEADER ---
PERFORM H3-HEADER 2 'Image'
*
PERFORM H3-RULE 0
*
PERFORM H3-HEADER 4 'left:'
*
COMPRESS GIF '/natw_sam.gif' INTO H3URL LEAVING NO
PERFORM H3-IMAGE H3URL 'NATweb left' 219 229 "L"
*
FOR II 1 TO 10
  PERFORM H3-LINE-BREAK
END-FOR
PERFORM H3-RULE 80
*
PERFORM H3-HEADER 4 'small right:'
*
COMPRESS GIF '/natw_sam.gif' INTO H3URL LEAVING NO
PERFORM H3-IMAGE H3URL 'NATweb small right' 100 100 'R'
*
FOR II 1 TO 5
  PERFORM H3-LINE-BREAK
END-FOR
*
PERFORM H3-RULE 0
*
PERFORM H3-TIME_DATE
*
* --- END HTML API ---
PERFORM H3-CLOSE-HTML
* --- END W3 PROCESSING ---
PERFORM W3END ##RPC
*
END
```

2. Using an HTML Editor

There are two alternatives:

- Creating static pages (you only enter HTML, which will be converted to a Natural subprogram)
- Creating dynamic pages (you enter HTML plus Natural program code).

You can, of course, also create pages that are partly dynamic, partly static.

Example: Creating Static Pages

```
<HTML>
<TITLE>NATweb - Test</TITLE>
<BODY bgColor=d3d3d3 >
<BR>
<center>
<h2>
This Natural subprogram was generated by a HTML page.
</h2>
</CENTER>
</BODY></HTML>
```

This Natural subprogram will be generated from the above HTML page:

```
* ----- SUBPROGRAM generated out of file:
* ----- C:\static.htm
DEFINE DATA
PARAMETER USING W3PARM
LOCAL USING W3CONST
LOCAL
* ----- PRIVATE VARIABLES -----
1 W3VALUE        (A250)
END-DEFINE
*
* ----- ERROR HANDLER -----
ON ERROR
  PERFORM W3ERROR ##W3ERROR
  PERFORM W3END ##RPC
  ESCAPE ROUTINE
END-ERROR
* ----- INITIALISE HTTP API -----
PERFORM W3INIT ##RPC
* ----- HEADER FOR SERVER -----
PERFORM W3CONTENT-TYPE 'text/html'
*
PERFORM W3TEXTLINE '<HTML>'
PERFORM W3TEXTLINE '<TITLE>NATweb - Test</TITLE>'
PERFORM W3TEXTLINE '<BODY bgColor=d3d3d3 >'
PERFORM W3TEXTLINE '<BR>'
PERFORM W3TEXTLINE '<center>'
PERFORM W3TEXTLINE '<h2>'
PERFORM W3TEXTLINE 'This Natural subprogram was generated by a HTML page.'
PERFORM W3TEXTLINE '</h2>'
PERFORM W3TEXTLINE '</CENTER>'
PERFORM W3TEXTLINE '</BODY></HTML>'
* ----- END HTTP API -----
PERFORM W3END ##RPC
* ----- END MAIN PROGRAM -----
*
* ----- SUBROUTINES -----
*
END
```

Example: Creating Dynamic Pages

```
<Natural><!--
 *
 * Read form Pers-View starting with value given by the
 * Parameter START
 *
 * Use HTML2NAT to generate a Natural Program
 *
 * 11.03.02
 *
 --></Natural>
 <! --- Variables to read the environment --->
 <Natural data><!--
 * ----- DATA -----
 1 H3VALUE        (A250)
 1 H3MAX            (I4)
 --></Natural>
 <! --- Head of the HTML page --->
 <HTML>
 <TITLE>Natural - Environment Test</TITLE>
```

```
<BODY bgColor=d3d3d3 >
<BR>
<center>
<h2>
This Natural subprogram was generated by a HTML page. The program had been
precompiled out of a HTML page.
<br><br>
</h2>
</center>
<br>
<hr>
<! --- Subprogram to write the output to work file,
       from where the server will read it --- >
<Natural DATA><!--
1 #CONTENT (A1/1:48)
1 REDEFINE #CONTENT
  2 #PERSONNEL-NUMBER  (N8)
  2 FILLER 1X
  2 #NAME              (A20)
  2 FILLER 1X
  2 #FIRST-NAME        (A15)
  2 FILLER 1X
  2 #AGE               (N2)
--></Natural>
<Natural SUB><!--
* ----- Do the OUTPUT -----
DEFINE SUBROUTINE WRITELINE
  PERFORM W3TEXT "<LI>"
*
  #PERSONNEL-NUMBER:=PERSONNEL-NUMBER
  #NAME:=NAME
  #FIRST-NAME:=FIRST-NAME
  #AGE:=AGE
  PERFORM W3HTMLARRAY #CONTENT(*) 48
*
  PERFORM H3-LINE-BREAK
END-SUBROUTINE
--></Natural>
<UL><PRE>
<! --- Parameter used for reading data from the DATABASE --->
<Natural DATA><!--
* ----- DATA -----
1 #VALUE (A20)
1 PERS-VIEW VIEW OF PERSONNEL
  2 PERSONNEL-NUMBER
  2 NAME
  2 FIRST-NAME
  2 AGE
--></Natural>
<! --- Main program to read the data --->
<Natural NOT>
<LI>Value1
<LI>Value2
<LI>...
</Natural>
<Natural><!--
* --- READ ENVIRONMENT ---
PERFORM W3READ-ENVIRONMENT ’START’ ’P’ H3VALUE H3MAX
IF H3MAX GT 0 THEN
  #VALUE := H3VALUE
ELSE
  #VALUE := "A"
```

```
END-IF
*
* ----- MAIN -----
F. FIND (100) PERS-VIEW NAME > #VALUE
  IF NO
    COMPRESS 'Sorry nothing found for:' #value '!' INTO H3VALUE
    PERFORM W3HTMLLINE H3VALUE
  END-NOREC
  IF *NUMBER > 0
    PERFORM WRITELINE
  END-IF
END-FIND
*
IF *NUMBER(F.) > 0
  PERFORM H3-RULE 0
*
  COMPRESS 'well done for: ' #value '!' ##HTTP_END INTO H3VALUE
  PERFORM W3HTMLLINE H3VALUE
END-IF
--></Natural>
</PRE></UL>
<! --- The footer of the HTML page --- >
<hr>
<BR>
<center>
<A HREF="index.htm">back to Index</A>
This program has been generated.
<Natural><!--
PERFORM H3-TIME_DATE
--></Natural>
</P>
</CENTER>
</BODY></HTML>
```

This Natural subprogram will be generated from the above HTML page:

```
* ----- SUBPROGRAM generated out of file:
* ----- C:\doit.htm
DEFINE DATA
PARAMETER USING W3PARM
LOCAL USING W3CONST
LOCAL
* ----- DATA -----
1 H3VALUE        (A250)
1 H3MAX            (I4)
1 #CONTENT (A1/1:48)
1 REDEFINE #CONTENT
  2 #PERSONNEL-NUMBER  (N8)
  2 FILLER 1X
  2 #NAME              (A20)
  2 FILLER 1X
  2 #FIRST-NAME        (A15)
  2 FILLER 1X
  2 #AGE               (N2)
* ----- DATA -----
1 #VALUE (A20)
1 PERS-VIEW VIEW OF PERSONNEL
  2 PERSONNEL-NUMBER
  2 NAME
  2 FIRST-NAME
  2 AGE
* ----- PRIVATE VARIABLES -----
```

```
1 W3VALUE        (A250)
END-DEFINE
*
* ----- ERROR HANDLER -----
ON ERROR
  PERFORM W3ERROR ##W3ERROR
  PERFORM W3END ##RPC
  ESCAPE ROUTINE
END-ERROR
* ----- INITIALISE HTTP API -----
PERFORM W3INIT ##RPC
* ----- HEADER FOR SERVER -----
PERFORM W3CONTENT-TYPE 'text/html'
*
* ----- MAIN PROGRAM -----
*
* Read form Pers-View starting with value given by the
* Parameter START
*
* Use HTML2NAT to generate a Natural Program
*
* 11.03.2002
*
PERFORM W3TEXTLINE '<! --- Variables to read the environment --->'
PERFORM W3TEXTLINE '<! --- Head of the HTML page --->'
PERFORM W3TEXTLINE '<HTML>'
PERFORM W3TEXTLINE '<TITLE>Natural - Environment Test</TITLE>'
PERFORM W3TEXTLINE '<BODY bgColor=d3d3d3 >'
PERFORM W3TEXTLINE '<BR>'
PERFORM W3TEXTLINE '<center>'
PERFORM W3TEXTLINE '<h2>'
PERFORM W3TEXTLINE 'This Natural subprogram was generated by a HTML page. Th'
 -'e program had been'
PERFORM W3TEXTLINE 'precompiled out of a HTML page.'
PERFORM W3TEXTLINE '<br><br>'
PERFORM W3TEXTLINE '</h2>'
PERFORM W3TEXTLINE '</center>'
PERFORM W3TEXTLINE '<br>'
PERFORM W3TEXTLINE '<hr>'
PERFORM W3TEXTLINE '<! --- Subprogram to write the output to work file'
PERFORM W3TEXTLINE '       from where the server will read it --- >'
PERFORM W3TEXTLINE '<PRE>'
PERFORM W3TEXTLINE '<! --- Parameter used for reading data from the'
 -' DATABASE --->'
PERFORM W3TEXTLINE '<! --- Main Program to read the data --->'
* --- READ ENVIRONMENT ---
PERFORM W3READ-ENVIRONMENT 'START' 'P' H3VALUE H3MAX
IF H3MAX GT 0 THEN
  #VALUE := H3VALUE
ELSE
  #VALUE := "A"
END-IF
*
* ----- MAIN -----
F. FIND (100) PERS-VIEW NAME > #VALUE
  IF NO
    COMPRESS 'Sorry nothing found for:' #value '!' INTO H3VALUE
    PERFORM W3HTMLLINE H3VALUE
  END-NOREC
  IF *NUMBER > 0
    PERFORM WRITELINE
  END-IF
```

```
END-FIND
*
IF *NUMBER(F.) > 0
  PERFORM H3-RULE 0
*
  COMPRESS 'well done for: ' #value '!' ##HTTP_END INTO H3VALUE
  PERFORM W3HTMLLINE H3VALUE
END-IF
PERFORM W3TEXTLINE '</PRE>'
PERFORM W3TEXTLINE '<! --- The footer of the HTML page --- >'
PERFORM W3TEXTLINE '<hr>'
PERFORM W3TEXTLINE '<BR>'
PERFORM W3TEXTLINE '<center>'
PERFORM W3TEXTLINE '<A HREF="index.htm">back to Index</A>'
PERFORM W3HTMLLINE 'This program has been generated.'
perform H3-TIME_DATE
PERFORM W3TEXTLINE '</P>'
PERFORM W3TEXTLINE '</CENTER>'
PERFORM W3TEXTLINE '</BODY></HTML>'
* ----- END HTTP API -----
PERFORM W3END ##RPC
* ----- END MAIN PROGRAM -----
*
*
* ----- SUBROUTINES -----
* ----- Do the OUTPUT -----
DEFINE SUBROUTINE WRITELINE
  PERFORM W3TEXT "<LI>"
*
  #PERSONNEL-NUMBER:=PERSONNEL-NUMBER
  #NAME:=NAME
  #FIRST-NAME:=FIRST-NAME
  #AGE:=AGE
  PERFORM W3HTMLARRAY #CONTENT(*) 48
*
  PERFORM H3-LINE-BREAK
END-SUBROUTINE
END
```

General Programming Considerations

Constant Values in the Local Data Area W3CONST

The local data area W3CONST contains a number of constant values which you might find useful:

##HTTP_NEWLINE, ##HTTP_NEWLINE_LENGTH and ##HTTP_NEWLINE_END

If you enter the ##HTTP_NEWLINE string into your HTML, you can use the subroutines W3TEXT and W3TEXT-ARRAY in the library SYSWEB to create a physical new line by compressing #HTTP_NEWLINE into the string. If you do not wish trailing blanks to be stripped, you can use ##HTTP_NEWLINE_END. ##HTTP_NEWLINE_LENGTH returns the length of the ##HTTP_NEWLINE string. ##HTTP_NEWLINE_END is always one character longer.

##HTTP_END

If you enter this string into your HTML, all routines which write their output to the return page will remove trailing blanks. If you do not wish trailing blanks to be removed, compress a ##HTTP_END string at the end of your string. There will be more output processing as a result.

##W3ERROR

Parameter used for calling W3ERROR.

##HTML_LT

Constant HTML value for "less than" sign (<).

##HTML_GT

Constant HTML value for "greater than" sign (>).

##HTML_AMP

Constant HTML value for "ampersand" sign (&).

##HTML_QUOT

Constant HTML value for "double quote" sign (").

##HTML_REG

Constant HTML value for "Registered Trademark" sign.

##HTML_COPY

Constant HTML value for "copyright" sign.

##HTML_NBSP

Constant HTML value for "no page breaking" space (’ ’).

Variables Defined by Value

All input variables are defined BY VALUE, that is, every value which is MOVE compatible can be used, especially strings.

Creating a Next Page

If your output possibly exceeds the limits of your return page, use the subroutine W3COUNTER in the library SYSWEB to evaluate how many bytes are free in the return page.

Changing the Amount of Data Transferred

To change the amount of data transferred between browser and Natural, go through the following steps:

1. Set the following variables and parameter to the same value:
   - the upper bound of the variable ##HTTP_RETURN_PAGE in the global data area W3CONTEX;
   - the upper bound of the variable RETURN_PAGE in the parameter data area W3PARM;
   - parameter RPC_INOUT_LENGTH in the `.ini` file of the used Natural Web Server Extension program. This defines the maximum length of a generated page.
2. Set the following variables to the same value:
   - the upper bound of the variable ##HTTP_ENVIRONMENT in the global data area W3CONTEX;
   - the upper bound of the variable ENVIRONMENT in the parameter data area W3PARM.
     This defines the maximum length of the data received. This value must be smaller than or equal to the maximum length of a generated page (defined in Step 1).
3. Recatalog W3CONTEX, W3PARM and W3ACCES in library SYSWEB (W3ACCES encapsulates all calls to W3CONTEX).
4. Recatalog all subprograms which are to be called via the Natural Web Server Extension and all programs called NAT-*, E3*, D3*, D4*and Web* in the library SYSWEB.

Testing Subprograms

There are three ways to test your subprograms:

1. Call the subprogram from your web browser.
2. Call the subprogram NAT-DIR in the library SYSWEB to see the contents of a Natural library. You can also specify the name of the library in the parameters, for example http://.../sysweb/ NAT-DIR?LIB=SYSEXT. Click on your program to start it.
3. If you do not want to call your subprogram from the web, you can use the Natural program WEB-ONL to simulate a remote call. The output of this program will be saved as a Natural text object. This "online execution" allows you to use the Natural Debugger.

The Natural Web Server Extensions

The Natural Web Server Extension is called from a HTTP server. The program repackages the parameters it receives from the HTTP server and performs an Entire Broker RPC or a DCOM call to the specified Natural subprogram or method.

Parameters

Data sent by the HTTP server is recognized and preprocessed. The URL, which was transmitted to the HTTP server in a URL-decoded (modified) form, is reset to its original state. All non-binary data can be transmitted as data and will be converted from ASCII to EBCDIC and vice versa, if necessary.

Initialization File

Only variables specified in your HTML page will automatically be transferred to the subprogram called. All other variables to be transferred must be specified in an ENV= entry of the .ini file. In this way, it is possible to add variables which will be treated as system environment variables. To add a system environment variable, specify a SETENV= entry in the `.ini` file.

Example .ini file

```
ENV=HTTP_REFERRER
  ENV=HTTP_HOST
  ;
  SETENV=VERSION:=alpha
  SETENV=BROKER:=local
```

Error Logging

To save the last HTML page that was transmitted from the server to a file, specify the TRACE_FILE parameter in your configuration file.

To return an error log, specify the ERROR_LOG_FILE parameter as log-file name in your configuration file.

To get your own error screen, specify the ERROR_TEMPLATE parameter in your configuration file with your desired HTML error page’s name. Environment variables can be specified in the HTML error page by using the prefix "$". With the environment variable $NWW_ENVIRONMENT, all environment variables transmitted to the subroutine called will be written as comment lines to the error page.

Data Areas

The subprograms in the Natural server environment have to use the parameter data area W3PARM from the library SYSWEB.

```
DEFINE DATA
  PARAMETER USING W3PARM
  ...
  END-DEFINE
```

The size of two specific variables in the global data area W3CONTEX has to be equal to the size of two specific variables in the parameter data area W3PARM:

GDA W3CONTEX	PDA W3PARM
##HTTP_RETURN_PAGE(A250/1:a)	##RPC.RETURN_PAGE(A250/1:a)
##HTTP_ENVIRONMENT(A250/1:b)	##RPC.ENVIRONMENT(A250/1:b)

The size of the variable ##RPC_RETURN_PAGE in the parameter data area W3PARM has to be equal to the value specified in RPC_INOUT_LENGTH defined in the configuration file of the calling program.

Naming Conventions of the Library SYSWEB

Subroutines W3*

W3* subroutines access the interface to your HTTP server in the Natural Web Server Extension. Such an interface consists (basically) of a parameter data area and of a log of the data transmitted. The W3* subroutines used by the subprogram are called by the HTTP server using the Natural RPC.

Subroutines H3*

If you call one the H3* subroutines from one of your subroutines, it creates a basic HTML tag.

Subprograms NAT*

There are some utilities to be called from the Internet.

Natural Text Members T3*

They describe what the Library SYSWEB contains, what the subroutine names are and which parameters can be passed. They also provide a code sample of how to invoke them. Use the utility nat-docu to access this online documentation.

Subprograms E3*

Sample code of the online documentation.

Members D3* and D4*

They are two demonstration applications.

Programs Web*

They are utilities that run from the Natural NEXT prompt.